Very good! Passed.

Computational Statistics Lab3

Group 6 - Araya Eamrurksiri & Oscar Pettersson

February 19, 2016

Assigment1: Cluster sampling

An opinion pool is assumed to be performed in several locations of Sweden by sending interviewers to this location. Of course, it is unreasonable from the financial point of view to visit each city. Instead, a decision was done to use random sampling without replacement with the probabilities proportional to the number of inhabitants of the city to select 20 cities. Explore the file ***population.xls****. Note that names in bold are counties, not cities.*

1.1 Import necessary information to R.

1.2 Use uniform random number generator to create a function that selects 1 city from the whole list by the probability scheme offered above (do not use standard sampling functions present in R).

A function is created for randomly selecting a municipality. Each municipality has a probability of being selected that is the ratio of its number of citizens and the total number of citizens in Sweden (9,340,682). We then get a CDF by "stacking" these probabilities on top of each other. By using the output of a uniform (0,1) random number generator and compare this to the first city for which the cumulative probability is at least this high, we choose a pseudo-random municipality from the 290 in the data set.

```
myfunc <- function(data){
    prob <- data$Population / sum(data$Population)
    if(sum(prob) != 1) stop

    set.seed(244)
    randnum <- runif(1)
    accum <- 0
    i <- 1
    while(accum <= randnum){
        accum <- accum + prob[i]
        i <- i + 1
    }
    count <- i - 1 #get the city index
    return(count)
}
```

1.3 Use the function you have created in step 2 as follows:

a. Apply it to the list of all cities and select one city
b. Remove this city from the list
c. Apply this function again to the updated list of the cities
d. Remove this city from the list
e. .. and so on until you get exactly 20 cities.

```
result <- NULL
for(i in 1:20){
    ind <- myfunc(data)
    result <- rbind(result,data[ind,])
    data <- data[-ind,]
}
```

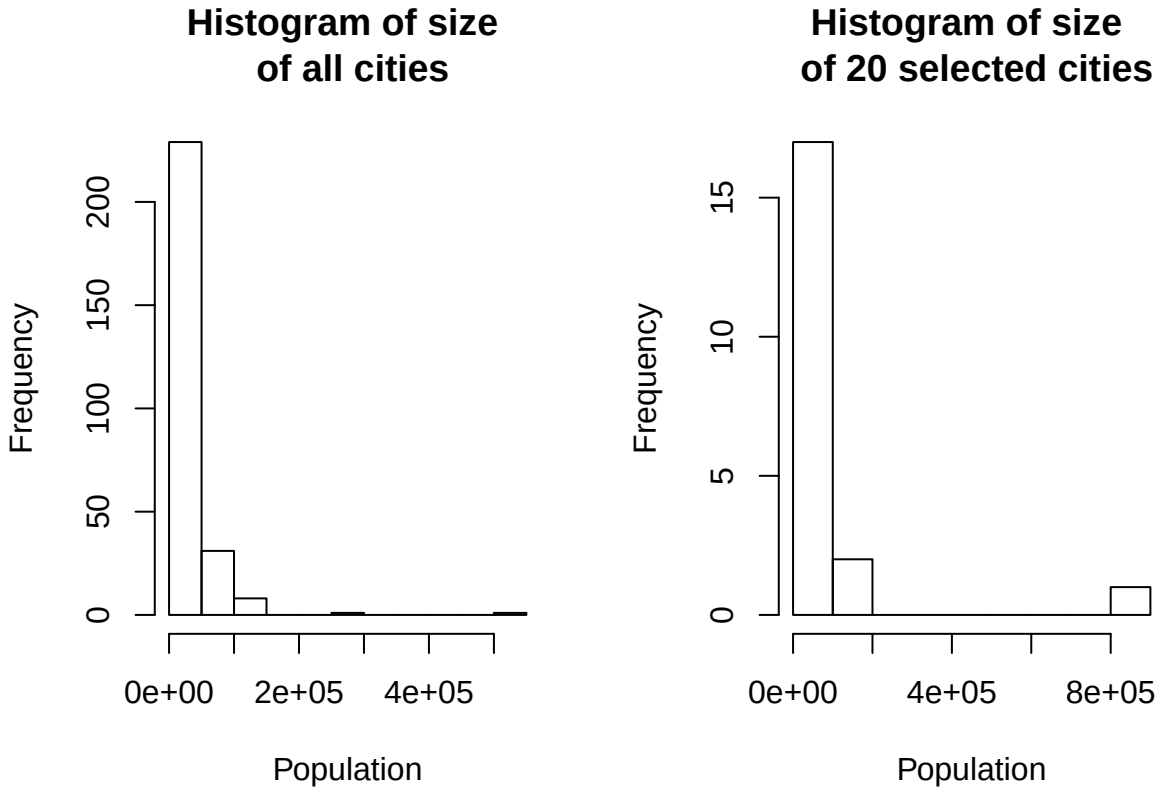
1.4 Run the program. Which cities were selected? What can you say about the size of the selected cities?

Following are the the 20 selected cities:

```
##    County.Municipality Population  density
## 1            Södertälje      85270 moderate
## 2                 Täby      63014 moderate
## 3       Upplands Väsby      38641 moderate
## 4         Upplands-Bro      23202 moderate
## 5                Tyresö      42602 moderate
## 6               Vaxholm      11001 moderate
## 7                Värmdö      37756 moderate
## 8            Vallentuna      29361 moderate
## 9             Österåker      39173 moderate
## 10             Enköping      39360 moderate
## 11                 Heby      13355 moderate
## 12                 Håbo      19452 moderate
## 13           Sundbyberg      37722 moderate
## 14                Tierp      20044 moderate
## 15              Uppsala     194751     high
## 16            Stockholm     829417     high
## 17           Norrköping     129254     high
## 18             Gislaved      29212 moderate
## 19               Gnosjö       9536    small
## 20                 Habo      10674 moderate
```

It can be seen that cities are randomly selected from the list. The size of these cities are mixed with high population (more than 100,000), moderate population (between 10,000 and 100,000) and low population (less than 10,000). After running this program for several times, it is noticed that the majority of cities have moderate population. Moreover, at least four cities that have high population are selected for each run and cities with low population are chosen once or twice.

1.5 Plot one histogram showing the size of all cities of the country. Plot another histogram showing the size of the 20 selected cities. Conclusions?



These two histograms look very much alike. Thus, we can conclude that getting an opinion poll from these 20 selected cities is likely to give a good representation as sending interviewers to visit every city in Sweden.

Assignment 2: Different distributions

2.1 Write a code generating double exponential distribution DE(0,1) from U(0,1) by using inverse CDF method. Explain how you obtained that code step by step. Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

The Laplace distribution has the probability density function (PDF)

$f(x) = \frac{\alpha}{2} \cdot e^{-\alpha \cdot |x-\mu|}$

In order to obtain the cumulative distribution function, CDF, we can divide this into two cases:

If $x - \mu < 0 \Leftrightarrow x < \mu$:

$f(x) = \frac{\alpha}{2} \cdot e^{\alpha \cdot (x-\mu)} \Rightarrow F(x) = \frac{\alpha}{2} \cdot \lim_{b \to -\infty} \int_b^t e^{\alpha \cdot (t-\mu)} dt = \frac{\alpha}{2} \cdot lim_{b \to -\infty} \left[\frac{e^{\alpha(t-\mu)}}{\alpha} \right]_b^x = \frac{1}{2} \cdot lim_{b \to -\infty} \left(e^{\alpha(x-\mu)} - e^{\alpha(b-\mu)} \right)$
$= \frac{1}{2} e^{\alpha(x-\mu)}$

Because of the limitations in x, $F(x)$ is bounded inside $]0, 0.5[$.

The inverse CDF is given by $y = F(x) \Leftrightarrow F^{-1}(y) = \mu + \frac{ln(2y)}{\alpha}$ for $y \in]0, 0.5[\,, \alpha \neq 0$

If $x - \mu \geq 0 \Leftrightarrow x \geq \mu$:

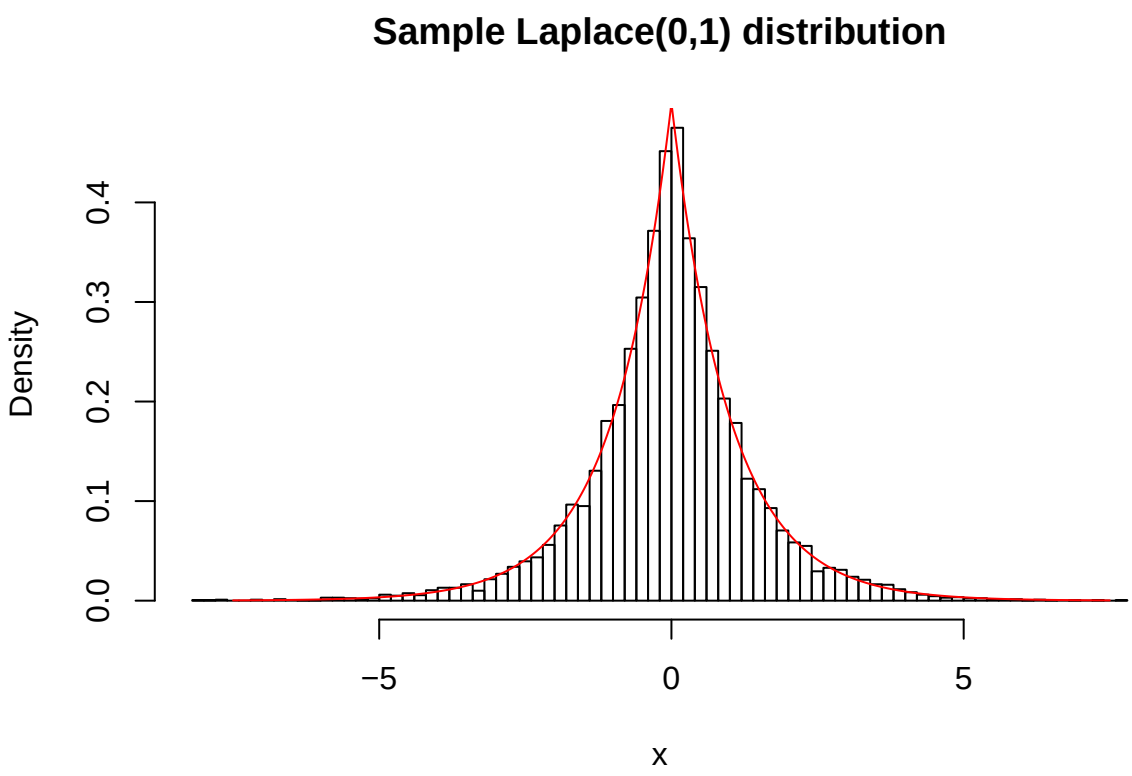
$f(x) = \frac{\alpha}{2} \cdot e^{-\alpha(x-\mu)} \Rightarrow F^{'}(x) = \frac{\alpha}{2} \cdot \int_{\mu}^{t} e^{-\alpha \cdot (t-\mu)} dt = \frac{\alpha}{2} \cdot \left[\frac{e^{-\alpha(t-\mu)}}{-\alpha} \right]_{\mu}^{x} = -\frac{1}{2} \cdot \left(e^{\alpha(x-\mu)} - e^{0} \right) = \frac{1}{2} - \frac{1}{2} e^{-\alpha(x-\mu)}$

Since we know that the total CDF for $x \in]-\infty, \mu[$ is 0.5, we let $F(x) = \frac{1}{2} + F^{'}(x)$. Now, $F(x)$ is bounded inside $[0.5, 1[$.

The inverse CDF is given by $y = F(x) \Leftrightarrow F^{-1}(y) = \mu - \frac{ln(2(1-y))}{\alpha}$ for $y \in [0.5, 1[\,, \alpha \neq 0$

The function below simulates from the Laplace distribution by first simulating u from $U(0,1)$. If $u >= 0.5$, then $x \geq \mu$ and the second inverse CDF function above is used. Otherwise, the first one is used.

```
Laplace_CDF_inv <- function(mu, alpha, U) {
  ifelse(test = U >= 0.5,
         yes = mu - log(2 * (1 - U)) / alpha,
         no = mu + log(2 * U) / alpha)
}

x <- Laplace_CDF_inv(mu = 0, alpha = 1, U = runif(1E4))
x_theo <- seq(from = -7.5, to = 7.5, by = 0.01)
f <- exp(-abs(x_theo)) / 2
```



Above, we see the 10,000 samples and the pdf of the Laplace distribution. The distribution looks very similar to a theoretical Laplace distribution.

2.2 Use Acceptance/rejection method with DE(0,1) as majorizing density to generate N(0,1) variables. Explain step by step how this was done. How did you choose constant c in this method? Generate 2000 random numbers N(0,1) using your code and plot the histogram. Compute the average rejection rate R in the acceptance/rejection procedure. What is the expected rejection rate ER and how close is it to R? Generate 2000 numbers from N(0,1) using standard rnorm procedure, plot the histogram and compare the obtained two histograms.

We will now try to generate random standard normal variables. Because we cannot obtain its CDF we cannot use the inverse CDF method and we will instead use the acceptance/rejection method using the Laplace($\mu = 0, \alpha = 1$) distribution, which is also symmetrical, as a majorizing function.

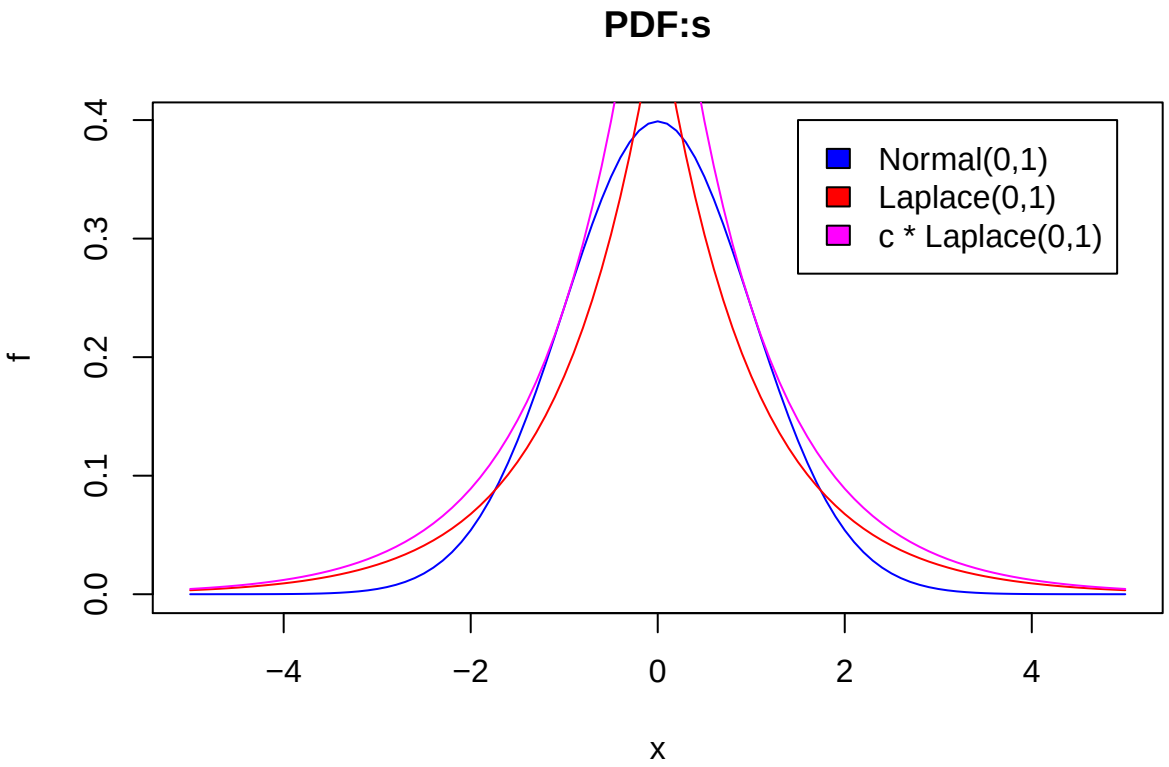
So, we have $f_x(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ and $f_y(x) = \frac{\alpha}{2} e^{-\alpha|x-\mu|}$.

Using the function from before, we generate Laplace distributed variables Y. Then, $runif()$ is used to get random U(0,1) variables U. We then investigate whether $U \leq \frac{f_x(Y)}{c \cdot f_y(Y)}$ and if that is true, Y is added to the pool of normal variables. We then go on until we have enough observations. So how do we first find the value of c?

We want c so that $c \cdot f_y(x) \geq f_x(x) \forall x$, so $c \geq \frac{f_x(x)}{f_y(x)} = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}}{\frac{\alpha}{2} e^{-\alpha|x-\mu|}} = \frac{2}{\alpha\sqrt{2\pi}} e^{-\frac{x^2}{2} + \alpha|x-\mu|}$

This has a maximum of about 1.315 for $x = \pm 1$, given that $\mu = 0$ and $\alpha = 1$.

Below, we see the pdf:s of the normal distribution ($f_x \sim N(0,1)$, blue), the Laplace distribution ($f_y \sim DE(0,1)$, red) and $c \cdot f_y$. We see that with our c-value, $c \cdot f_y \geq f_x$ for any value of x.



Now, with $c \approx 1.315$ we have a high enought value of c. Using the acceptance/rejection method, we can expect that the acceptance rate is the reciprocal of c. So, we probably need more than 2,600 iterations. (Though, *here* the loop is worked around because of R:s efficiency with vectors.)

```
print(c)
```

```
## [1] 1.315489
```

```
print(1 - 1 / c) # Theoretical rejection rate
```

```
## [1] 0.2398265
```

```
print(2000 / (1 / c)) # Estimated number of "iterations" needed
```

```
## [1] 2630.978
```

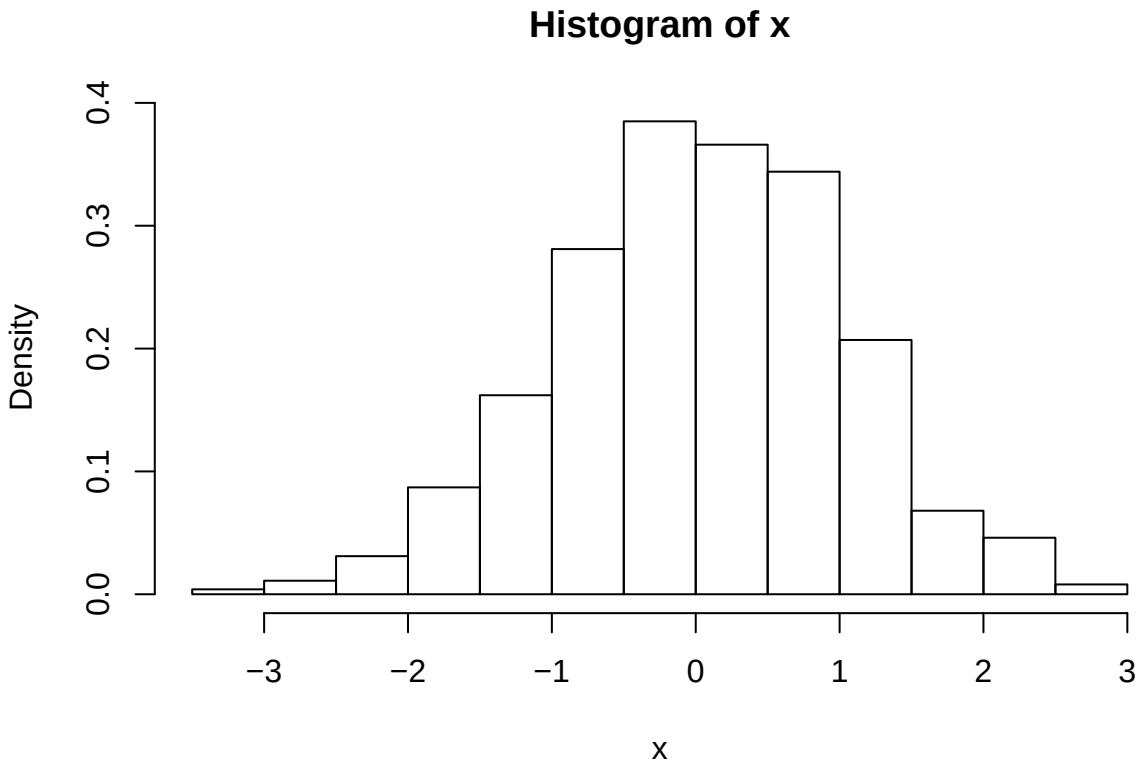
```
set.seed(12345)
Y <- Laplace_CDF_inv(mu = 0, alpha = 1, U = runif(5E3)) # 5,000 Laplace(0,1) variables
U <- runif(n = 5E3, min = 0, max = 1) # 5,000 U(0,1) variables
x_temp <- ifelse(U <= f_nrm(Y) / (c * f_lpl(mu = 0, alpha = 1, x = Y)), yes = Y, no = NA)

index <- which(cumsum(!is.na(x_temp)) == 2000)[1]
x <- x_temp[1:index] # Keep the first 2,000 good observations and the NA:s in between

print(length(x)) # Actual number of "iterations" needed
```

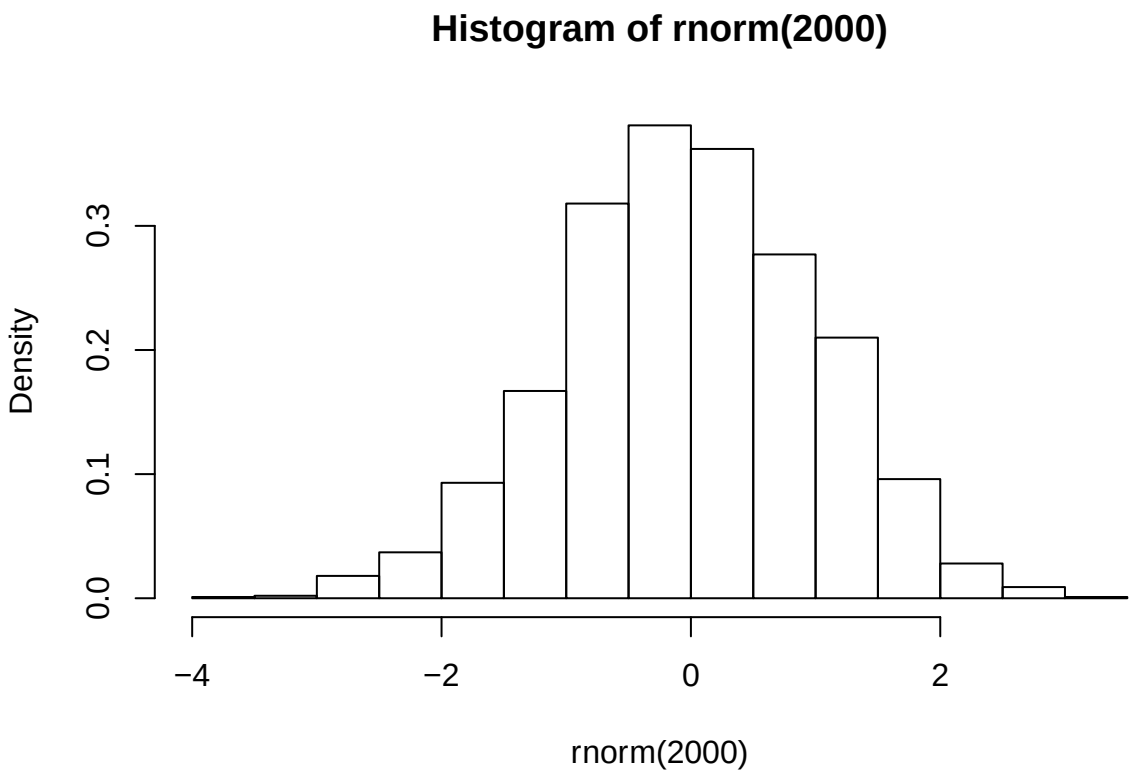
```
## [1] 2639
```

```
print((length(x) - 2000) / length(x)) # Actual rejection rate
```

```
## [1] 0.2421372
```

So, we have got 2,000 samples and we needed to create 2639 sample pairs of Y and U to get them. So the acceptance (or rejection) rate and the total number of iterations are indeed close to what we could expect.





Looking at the histograms we can see that our sampled distribution seems to have a distribution close to the (pseudo-randomised) normal.

Appendix

Contribution

We have been discussing and noticed that most of the solutions are very similar. The code and most of the text for the first assignment is taken from Araya and the code and most of the text for the second one is from Oscar.

R-code

```
library(XLConnect)
wb <- loadWorkbook("/Users/lynn/Documents/LiU/732A38 Computational statistics/Lab3/population2.xls")
pop <- readWorksheet(wb, sheet="Table")

#manage dataset
pop <- pop[,1:3]
data <- pop[which(nchar(pop$Code) > 2),] #290 Cities
myfunc <- function(data){
    prob <- data$Population / sum(data$Population)
    if(sum(prob) != 1) stop

    set.seed(244)
    randnum <- runif(1)
```

```
    accum <- 0
    i <- 1
    while(accum <= randnum){
        accum <- accum + prob[i]
        i <- i + 1
    }
    count <- i - 1 #get the city index
    return(count)
}
result <- NULL
for(i in 1:20){
    ind <- myfunc(data)
    result <- rbind(result,data[ind,])
    data <- data[-ind,]
}
result$density <- rep("moderate",20)
result[which(result$Population < 10000),]$density <- "small"
result[which(result$Population > 100000),]$density <- "high"
rownames(result) <- NULL
result[,2:4]
par(mfrow=c(1,2))
hist(data$Population, xlab="Population", main="Histogram of size \n of all cities")
hist(result$Population, xlab="Population", main="Histogram of size \n of 20 selected cities")
Laplace_CDF_inv <- function(mu, alpha, U) {
  ifelse(test = U >= 0.5,
         yes = mu - log(2 * (1 - U)) / alpha,
         no = mu + log(2 * U) / alpha)
}

x <- Laplace_CDF_inv(mu = 0, alpha = 1, U = runif(1E4))
x_theo <- seq(from = -7.5, to = 7.5, by = 0.01)
f <- exp(-abs(x_theo)) / 2
hist(x, 100, main = "Sample Laplace(0,1) distribution", probability = TRUE)
points(f ~ x_theo, type="l", xlab = "x", main = "Theoretical PDF for Laplace(0,1) distribution", col="r
mu <- 0
alpha <- 1

f_nrm <- function(x) exp(-0.5 * x ^ 2) / sqrt(2 * pi)
f_lpl <- function(mu, alpha, x) alpha * exp(-alpha * abs(x - mu)) / 2

grid <- seq(from = -5, to = 5, by = 0.1)
fx <- f_nrm(grid)
fy <- f_lpl(mu = 0, alpha = 1, x = grid)
plot(fx ~ grid, col = "blue", type = "l", xlab="x", ylab="f", main ="PDF:s")
legend(x = 1.5, y = 0.4, legend = c("Normal(0,1)", "Laplace(0,1)", "c * Laplace(0,1)"), fill = c("blue"
points(fy ~ grid, col = "red", type = "l")
c <- 2 / (alpha * sqrt(2 * pi)) * exp(-.5 + alpha * abs(1 - mu))
cfy <- c * fy
points(cfy ~ grid, col = "magenta", type = "l")
print(c)
print(1 - 1 / c) # Theoretical rejection rate
print(2000 / (1 / c)) # Estimated number of "iterations" needed
set.seed(12345)
```

```r
Y <- Laplace_CDF_inv(mu = 0, alpha = 1, U = runif(5E3)) # 5,000 Laplace(0,1) variables
U <- runif(n = 5E3, min = 0, max = 1) # 5,000 U(0,1) variables
x_temp <- ifelse(U <= f_nrm(Y) / (c * f_lpl(mu = 0, alpha = 1, x = Y)), yes = Y, no = NA)

index <- which(cumsum(!is.na(x_temp)) == 2000)[1]
x <- x_temp[1:index] # Keep the first 2,000 good observations and the NA:s in between

print(length(x)) # Actual number of "iterations" needed
print((length(x) - 2000) / length(x)) # Actual rejection rate
hist(x, probability = TRUE)
hist(rnorm(2E3), probability = TRUE)
## NA
```